Colonizzare la noosfera

.

Colonizzare la noosfera

di Eric S. Raymond

Aprile 1998

Dopo aver osservato la contraddizione esistente tra l'ideologia “ufficiale” definita dalle licenze open source e il comportamento degli hacker, vengono prese in esame gli usi che in concreto regolano la proprietà e il controllo del software open source. Si scopre così che queste implicano una teoria sottostante dei diritti di proprietà praticamente omologa alla teoria di Locke sulla proprietà terriera. Si mette poi in relazione ciò con l'analisi della cultura hacker in quanto “cultura del dono”, dove i partecipanti competono per il prestigio di regalare tempo, energia, creatività. Per passare infine a considerare le implicazioni di tale analisi riguardo la risoluzione dei conflitti nella cultura in generale, sviluppando alcune utili indicazioni generali.

Sommario:

1. Una contraddizione introduttiva

Chiunque abbia occasione di osservare per un po' il mondo, frenetico e tremendamente produttivo, del software open source su Internet, non potrà fare a meno di notare un interessante contraddizione tra quello in cui gli hacker dicono di credere e il modo con cui in realtà agiscono – tra l'ideologia ufficiale della cultura open source e la sua pratica concreta.

Le culture sono macchine in via di adattamento. La cultura open source è la risposta a una specifica serie di impulsi e pressioni. Come sempre, l'adattamento della cultura alle circostanze si manifesta sia come ideologia cosciente sia come conoscenza implicita, cosciente o semi-cosciente. E, come spesso accade, gli aggiustamenti impliciti risultano parzialmente in contrasto con l'ideologia cosciente.

In questo saggio indagherò le radici di tale contraddizione, così da scoprire di quali impulsi e pressioni si tratti. Da qui si potranno trarre degli elementi interessanti sulla cultura hacker e sulle relative consuetudini. Concluderò suggerendo alcune modalità per meglio utilizzare la conoscenza implicita insita in tale cultura.

I riferimenti tra parentesi quadre si riferiscono alla bibliografia e alle note finali.

2. Le varietà dell'ideologia hacker

L'ideologia della cultura open source di Internet (ciò in cui gli hacker dicono di credere) è un argomento di per sé alquanto complesso. Tutti coloro che ne partecipano si dicono concordi sul fatto che l'open source (ovvero, il software in libera ridistribuzione e soggetto a pronta evoluzione e modifica come risposta alle esigenze in mutamento) sia qualcosa di positivo e degno degli sforzi collettivi. È sulla base di tale concordanza generale che si definisce efficacemente l'affiliazione in tale cultura. Tuttavia, quel che varia in maniera considerevole sono le motivazioni individuali e le relative sotttoculture.

Un primo livello di differenziazione è l'eccessivo fanatismo; ciò nel caso in cui lo sviluppo open source venga considerato sia soltanto come giusto mezzo per il raggiungimento di un determinato fine (buoni strumenti, giocattoli divertenti, un gioco interessante da fare) sia come scopo ultimo di per sé.

Una persona molto fanatica direbbe: “Il free software è tutto per me! Vivo solo per creare programmi e risorse utili e bellissime, e per diffonderli in giro.” Una persona moderatamente fanatica direbbe: “L'open source è una bella storia a cui ho deciso di dedicare parecchio del mio tempo per far sì possa concretizzarsi.” Una persona scarsamente fanatica direbbe: “Si, qualche volta l'open source è OK. Mi piace giocarci e rispetto quelli che l'hanno messo su.”

Un altro livello di variazione risiede nell'ostilità al software commerciale e/o a quelle aziende percepite come dominatrici del relativo mercato.

Una persona molto anticommerciale direbbe: “Il software commerciale è un furto e un'imposizione. Scrivo free software per porre fine a questa mostruosità.” Una persona moderatamente anticommerciale direbbe: “In generale il software commerciale è OK perché i programmatori meritano d'essere pagati, ma le aziende che s'arricchiscono con prodotti scadenti e la fanno da padroni sono una calamità.” Una persona non anticommerciale direbbe: “Il software commerciale è OK, io uso e/o scrivo free software soltanto perché questo mi piace di più.”

Tutte e nove le categorie possibili, risultanti dalla combinazione delle variazioni elencate, sono presenti nella cultura open source. Il motivo per cui è il caso di puntualizzare le distinzioni deriva dal fatto che ciascuna di esse persegue obiettivi diversi, e altrettanto diversi comportamenti di adattamento e cooperazione.

Storicamente la parte più visibile e più organizzata della cultura hacker è stata sia molto fanatica sia molto anticommerciale. La Free Software Foundation (FSF), fondata da Richard M. Stallman, fin dai primi anni '80 ha concretamente sostenuto lo sviluppo open source inclusa la realizzazione di programmi quali Emacs e GCC, tuttora strumenti di base per il mondo open source di Internet. Una predominanza che pare destinata a proseguire nel prossimo futuro.

Per molti anni la FSF è stato l'unico fulcro importante per l'hacking open source, producendo un numero enorme di strumenti ancor'oggi cruciali per l'intera cultura. Inoltre, la FSF è stata a lungo l'unico sponsor dell'open source dotato di identità istituzionale visibile per gli osservatori esterni alla cultura hacker. Da lì arriva l'efficace definizione di “free software”, un termine deliberatamente carico di valenze provocatorie

(valenze altrettanto deliberatamente evitate dalla più recente etichetta open source http://www.opensource.org).

Quindi, le percezioni della cultura hacker sia dall'interno sia dall'esterno tendevano a identificarla con l'approccio fanatico e gli obiettivi primari come anticommerciali, tipici della FSF (personalmente Richard Stallman nega di essere anticommerciale, ma il suo programma è stato in tal senso interpretato dalla maggior parte delle persone, inclusi i suoi più accaniti sostenitori). L'impulso vigoroso ed esplicito della FSF a "distruggere la costrizione del software!" è divenuto l'elemento più pregnante dell'ideologia hacker, e Richard Stallman il più vicino al ruolo di leader della cultura hacker.

I termini della licenza FSF, la cosiddetta "General Public Licence" (GPL), ben esprimono siffatta attitudine e restano ampiamente in voga nel mondo open source. Nel 1997 circa la metà del software presente su Sunsite, l'archivio più grande e più noto del mondo Linux, in North Carolina, era contrassegnato dai termini espliciti della licenza GPL.

Ma la FSF non è mai stata l'unica squadra in campo. Nella cultura hacker è sempre esistita un'ala più moderata, meno provocatoria e più vicina al mercato. I più pragmatici erano fedeli non tanto a un'ideologia quanto piuttosto a un gruppo di tradizioni informatiche basate sul lavoro dell'open source e antecedenti alla FSF. Fatto ancor più importante, tali tradizioni comprendevano le culture tecniche interconnesse di Unix e dell'Internet pre-commerciale.

L'approccio tipicamente pragmatico è anticommerciale solo in maniera moderata, e la sua maggiore lagnanza contro il mondo delle corporation non è la "costrizione" di per sé. Piuttosto, il perverso rifiuto da parte di tale mondo nell'adottare un approccio superiore incorporando Unix, standard aperti e software open source. Se il pragmatico odia qualcosa, è meno probabile si tratti dei "costrittori" in senso generale quanto piuttosto degli attuali dittatori dell'industria informatica, ieri IBM e oggi Microsoft.

Per i pragmatici, la GPL è importante in quanto strumento anziché come fine ultimo. Il suo maggior valore non sta tanto nel fatto di rappresentare un'arma contro la "costrizione", bensì in quanto strumento per incoraggiare la condivisione del software e la crescita delle comunità di sviluppo sul *modello a bazaar* http://www.apogeonline.com/openpress/doc/cathedral.html. Il pragmatico apprezza strumenti e giocattoli efficaci più di quanto disdegni la commercialità, e riesce a utilizzare anche software commerciale d'alta qualità senza alcun problema ideologico. Al contempo, l'esperienza dell'open source gli ha fatto apprendere standard tecnici di una qualità che ben pochi software "chiusi" possono eguagliare.

Per molti anni il punto di vista pragmatico si è espresso all'interno della cultura hacker essenzialmente con l'ostinato rifiuto ad accettare tout court la GPL in particolare o l'agenda di FSF in generale. Nel corso degli anni '80 e dei primi '90, tale atteggiamento veniva associato con i fan Unix di Berkeley, gli utenti della licenza BSD, e i primi tentativi di realizzare programmi Unix open source partendo dai sorgenti base di BSD. Tentativi che però non sono riusciti a costruire comunità bazaar di proporzioni significative, finendo col diventare decisamente frammentati e inefficaci.

Il pragmatismo non ebbe alcuna forza di base fino all'esplosione di Linux nel 1993-1994. Pur non essendosi mai posto in contrapposizione a Richard Stallman, Linus Torvalds ha stabilito un esempio considerando benignamente la crescita dell'ambito industriale legato a Linux, appoggiando pubblicamente l'utilizzo di software commerciale di alta qualità per compiti specifici, e deridendo gentilmente quegli elementi più puristi e fanatici all'interno della cultura hacker.

Effetto collaterale della rapida crescita di Linux fu l'iniziazione di un vasto numero di nuovi hacker che avevano giurato fedeltà a Linux, riconoscendo al programma della FSF un interesse storico. Anche se la nuova ondata di hacker Linux poteva descrivere il sistema operativo come "la scelta della generazione GNU", la maggior parte di loro tendeva a emulare Torvalds più che Stallman.

Gradatamente i puristi anticommerciali si ritrovarono ad essere una minoranza. Il cambiamento non venne però a galla fino all'annuncio nel febbraio 1998 di Netscape relativo alla distribuzione pubblica dei sorgenti di Navigator 5.0. Ciò risvegliò l'interesse del mondo delle corporation nel "free software". La

conseguente chiamata alla cultura hacker di approfittare di questa opportunità senza precedenti e di ridefinirne l'obiettivo da “free software” a “open source”, venne accolta da una tale e immediata approvazione che sorprese chiunque vi si trovasse coinvolto.

Nel successivo sviluppo rafforzativo, da metà anni '90 l'ala pragmatica della cultura andava trasformandosi in policentrica. Dalla radice Unix/Internet presero a germogliare altre comunità semi-indipendenti dotate di una propria autocoscienza nonché dei propri leader carismatici. Tra queste, la maggiore dopo Linux fu la cultura Perl sotto Larry Wall. Più piccole ma comunque significative, anche le enclave radunatesi intorno ai linguaggi Tcl di John Osterhout e Python di Guido Van Rossum. Tutte e tre queste comunità diedero corpo all'indipendenza ideologica, realizzando altrettanti schemi di licenza non-GPL.

3. Teoria promiscua, pratica puritana

Ciascuno di questi passaggi rimane comunque caratterizzato dalla teoria ampiamente condivisa su quel che s'intende con “free software” o “open source”. La formulazione più chiara di questa teoria si trova nelle varie licenze open source, ognuna delle quali include elementi centrali e comuni.

Nel 1997 tali elementi vennero distillati nelle linee-guida della Debian Free Software, a loro volta confluite nella *Open Source Definition* (OSD). In pratica si dice che la licenza open source deve proteggere il diritto incondizionato di ciascuno a modificare il software open source (e a redistribuirne le versioni modificate).

Quindi, la teoria implicita dell'OSD (e delle altre licenze a questa conformatesi, tipo GPL, BSD, e la licenza artistica di Perl) è che chiunque può modificare qualunque cosa. Nulla impedisce che una decina di persone diverse prendano un qualsiasi prodotto open source (ad esempio, il compilatore C gcc della Free Software Foundation), ne duplichino i sorgenti e li sviluppino in direzioni evolutive differenti, il tutto sempre sostenendo di star lavorando a un nuovo prodotto.

In pratica, però, tale “biforcazione” non avviene quasi mai. Le divaricazioni dei grossi progetti costituiscono un evento raro, e sono sempre accompagnate da ridefinizioni e ampie auto-giustificazioni pubbliche. È chiaro che, in casi quali la divisione Emacs/XEmacs in ambito GNU, oppure quella gcc/egcs, o ancora le varie scissioni dei gruppi BSD, ai “secessionisti” è parso di trasgredire una norma comunitaria assai radicata.

In realtà (e in contraddizione con la teoria condivisa secondo cui chiunque possa modificare qualunque cosa) la cultura open source possiede una serie di usi e costumi sulla proprietà, elaborati ma essenzialmente non dichiarati. Essi stabiliscono a chi e in quali circostanze sia consentito modificare il software, e (soprattutto) chi abbia il diritto di ridistribuire nuovamente le versioni modificate alla comunità.

I tabù di una cultura danno sostegno alle sue norme. Risulterà perciò utile riassumerne qui i più importanti.

- Esiste una forte pressione sociale contro i progetti divaricanti. Ciò non accade se non dietro la scusa della necessità impellente, fornendo ampie auto-giustificazioni pubbliche e sotto nuovo nome.
- È disapprovata la distribuzione delle modifiche apportate a un progetto senza la cooperazione dei moderatori, con l'eccezione di tipici casi particolari quali i minimi aggiustamenti relativi alle porte.
- La rimozione del nome di una persona dalla cronologia di un progetto, nonché dall'elenco di quanti hanno collaborato o l'hanno co-mantenuto, non avviene assolutamente senza esplicito consenso della medesima.

Nella parte restante di questo saggio, prenderemo dettagliatamente in esame tali tabù e le abitudini correnti in tema di proprietà. Vedremo non soltanto come funzionano, ma anche cosa rivelano riguardo

alle dinamiche sociali sottostanti e alle strutture d'incentivazione tipiche della comunità open source.

4. Proprietà e open source

Cosa s'intende con “proprietà” quando questa è replicabile all'infinito, altamente malleabile, e la cultura circostante non è dotata né di relazioni di potere coercitivo né dell'economia derivante dalla penuria di materiali?

In realtà, nel caso della cultura open source si tratta di una domanda dalla risposta facile. Soltanto il proprietario (o i proprietari) di un progetto detiene il diritto esclusivo, riconosciutogli dall'intera comunità, di *ridistribuire le versioni modificate*.

(Nella discussione sulla “proprietà” in questo capitolo, userò il singolare, come se ogni progetto fosse di proprietà di un solo individuo. È tuttavia ovvio che potrebbe anche trattarsi di gruppi di persone. Le dinamiche interne di tali gruppi verranno esaminate più avanti).

Secondo le licenze standard open source, tutte le entità coinvolte rivestono il medesimo ruolo nel gioco evolutivo. In pratica esiste però una distinzione molto ben riconosciuta tra gli aggiustamenti “ufficiali” – approvati e integrati nel software in evoluzione da parte di quanti, pubblicamente riconosciuti, si occupano della sua gestione – e quelli “volanti” realizzati da terze parti. In genere questi sono rari, e non riscuotono molta fiducia.

È facile comprendere perché la ridistribuzione pubblica si riveli una questione fondamentale. La convenzione incoraggia la gente a farsi le proprie “patch” per uso personale ogni volta che sia necessario. Le consuetudini mostrano indifferenza nei confronti di quanti ridistribuiscono versioni modificate all'interno di un gruppo chiuso di utenti o di sviluppatori. La proprietà diviene invece questione centrale soltanto quando le modifiche, in competizione con la versione originale, vengono diffuse nell'intera comunità open source.

In generale ci sono tre modi per acquisire la proprietà di un progetto open source. Uno, il più ovvio, è avviare la nascita del progetto. Se fin dall'inizio esso è stato mantenuto da una sola persona e questi è ancora attivo, la consuetudini non permettono neppure di *chiedere* a chi appartenga la proprietà del progetto.

La seconda maniera è ottenere la proprietà del progetto dal proprietario precedente (anche noto come “il passaggio del testimone”). È ben chiaro alla comunità che i proprietari hanno il dovere di passare i progetti a successori competenti, quando non possono o non vogliono più investire il tempo necessario nel lavoro di sviluppo o di mantenimento.

Risulta significativo che nel caso di grossi progetti, tali trasferimenti del controllo normalmente vengono annunciati con una certa enfasi. Mentre non sono noti casi di interferenza da parte della comunità open source sui successori scelti dai proprietari, la pratica convenzionale incorpora chiaramente la legittimazione pubblica come fatto importante.

Per progetti minori, generalmente è sufficiente includere il cambio di proprietà nella storia dei cambiamenti inclusa in calce a ogni progetto. Chiaramente si presume che se l'ex-proprietario non abbia passato il controllo volontariamente, lo riassuma con l'apporto della comunità dopo aver esposto le obiezioni in pubblico entro un ragionevole periodo di tempo.

Il terzo modo per acquisire la proprietà di un progetto è far presente che questo ha bisogno di lavori e il proprietario è scomparso o ha perso interesse. Chi volesse procedere, ha prima la responsabilità di cercare il proprietario. Se ciò risulta impossibile, allora va annunciato in un luogo ben visibile (tipo, il newsgroup di Usenet dedicato all'area di quell'applicazione) che il progetto sembra esser orfano, e che si vorrebbe assumerne la responsabilità.

Secondo la convenzione, occorre far trascorrere un certo periodo di tempo prima di procedere con l'auto-

dichiarazione di essere il nuovo proprietario. In questo intervallo, se qualcun altro annuncia di aver lavorato al progetto, tocca prima a lui. È considerata buona educazione rendere note pubblicamente e più volte le proprie intenzioni. Come pure farlo in molti forum rilevanti (newsgroup e mailing list attinenti). Ancor meglio se si dimostra pazienza in attesa di risposte. In generale, più sono visibili gli sforzi per consentire all'ex-proprietario di farsi avanti, e più l'acquisizione risulterà comprovata.

Una volta seguito questo iter in mancanza di obiezioni, sarà possibile dichiararsi proprietari del progetto orfano e annotare ciò nelle note. Tale processo risulta però meno sicuro del passaggio del testimone, e non ci si potrà considerare pienamente legittimati fino a quando non si siano apportati miglioramenti sostanziali per l'intera comunità.

Per vent'anni ho osservato il dipanarsi di queste consuetudini in azione, fin dall'epoca dell'antica storia pre-FSF del software open source. Vale la pena sottolinearne alcuni tratti caratteristici. Il più interessante è il fatto che la maggior parte degli hacker ha seguito tale processo senza rendersene granché conto. Non a caso quanto enunciato sopra potrebbe rivelarsi il primo tentativo cosciente e ragionevolmente completo di elaborarne un riassunto scritto.

Altro tratto degno di nota è il fatto che queste convenzioni inconsce sono state seguite con una coerenza notevole e perfino incredibile. Ho osservato personalmente l'evoluzione di, letteralmente, centinaia di progetti open source, e posso contare sulle dita le violazioni significative viste o riportate all'iter descritto sopra.

Terza caratteristica interessante è l'evoluzione di tali consuetudini nel corso del tempo, spostatesi sempre verso un'unica direzione. Ovvero quella di stimolare maggior responsabilità pubblica, maggior attenzione collettiva e maggior cura nel preservare i nomi dei partecipanti e le cronologie delle modifiche in ogni progetto, così da stabilire, tra l'altro, la legittimità dei proprietari correnti.

Queste caratteristiche sono la testimonianza della non casualità delle abitudini in uso, rivelandosi anzi queste il risultato di un qualche tipo di una pianificazione implicita o di un percorso generativo sviluppatosi all'interno della cultura open source che risulta assolutamente fondamentale alle modalità in cui si esprime.

Uno dei primi commenti ricevuti sottolineava come il contrasto tra la cultura hacker su Internet e quella dei cracker/pirati (il “warez d00dz" centrato sul gioco del “cracking” e le bulletin-board pirate) chiariva abbastanza bene i percorsi generativi di entrambe. Torneremo al “d00dz” come contrasto più avanti in questo saggio.

5. Locke e la proprietà terriera

Per aiutare la comprensione di questo percorso evolutivo, giova ricordare un'analogia storica che è ben lontana dalle comuni preoccupazioni degli hacker. Come suona forse familiare a chi ha studiato storia del diritto e filosofia politica, la teoria della proprietà implicata più sopra risulta virtualmente identica alla teoria del diritto consuetudinario sulla proprietà terriera d'origine angloamericano.

Secondo tale teoria, sono tre le modalità per acquisire la proprietà della terra.

In una zona di frontiera, dove esiste terra mai appartenuta a nessuno, è possibile acquisirne la proprietà tramite l'*insediamento*, ovvero lavorandola, cintandone i confini, difendendone l'atto di proprietà.

Il metodo comune per trasferirsi in aree colonizzate è il *trasferimento dell'atto di proprietà*, cioè ricevendo quest'ultimo dal proprietario precedente. In tale teoria è importante il concetto di “catena degli atti”. La prova ideale del diritto alla proprietà è la concatenazione degli atti precedenti e dei trasferimenti, rimandando all'epoca in cui quell'appezzamento venne colonizzato per la prima volta.

Infine, la teoria del diritto consuetudinario riconosce che l'atto in questione possa esser stato perso o abbandonato (nel caso, ad esempio, il proprietario sia deceduto senza eredi, o quando non sia più

possibile consultare gli archivi in grado di confermare la catena degli atti fino all'origine). Un appezzamento di terreno rimasto vacante può essere reclamato da qualcun altro che lo occupa, lo migliora, ne difende l'atto di proprietà come se vi si insediasse per la prima volta.

Al pari delle usanze degli hacker, questa teoria si è evoluta organicamente in un contesto dove l'autorità centrale era debole o inesistente. Si è sviluppata in un periodo di oltre mille anni a partire dal diritto tribale scandinavo e germanico. Essendo stata organizzata e razionalizzata agli albori dell'era moderna dal filosofo politico inglese John Locke, talvolta viene indicata come la teoria della proprietà terriera di Locke.

Ovviamente c'è stata la tendenza a sviluppare teorie analoghe laddove la proprietà rivestiva un notevole valore economico o di sopravvivenza, e dove non esisteva alcuna autorità abbastanza potente da costringere la centralizzazione delle merci ricercate. Ciò è vero perfino nelle culture dei cacciatori-raccoglitori, che vengono romanticamente ritenute prive del concetto di “proprietà”. Per esempio, la tradizione dei nativi !Kung San nel deserto del Kalahari (Africa australe) non prevede la proprietà dei territori di caccia. Ma *esiste* quella relativa alle pozze d'acqua e alle sorgenti, secondo una teoria assai simile a quella di Locke.

L'esempio dei !Kung San è indicativo poiché dimostra come le convenzioni sulla proprietà del tipo alla Locke nascano soltanto laddove i possibili proventi della risorsa in questione superino i costi previsti per difenderla. I territori di caccia non appartengono a nessuno perché i proventi della caccia risultano altamente variabili e imprevedibili, e pur essendo assai apprezzati, neppure costituiscono una necessità per la sopravvivenza quotidiana. Le pozze d'acqua, all'opposto, sono vitali per la sopravvivenza e relativamente piccole da difendere.

La “noosfera” indicata nel titolo di questo saggio è il territorio delle idee, lo spazio dove convivono tutte le concezioni possibili. [N]. È quindi possibile sostenere che le convenzioni degli hacker sulla proprietà non fanno altro che applicare la teoria di Locke all'ambito della noosfera, lo spazio di tutti i programmi. Ecco quindi spiegato il titolo, “colonizzare la noosfera”: esattamente quello che fa chiunque dia avvio a un nuovo progetto open source.

Fare Rideau giustamente sottolinea il fatto che gli hacker non operano proprio nel territorio dei puri concetti. Egli sostiene che gli hacker possiedono *i progetti della programmazione* – punti di fuoco intenzionali del lavoro materiale (sviluppo, servizi, etc.), ai quali vengono associati elementi quali reputazione, fiducia, etc. Di conseguenza, sempre secondo Fare Rideau, lo spazio occupato dai progetti degli hacker non è tanto la noosfera quanto piuttosto una sorta di suo doppio, lo spazio dei progetti di quei programmi che esplorano la noosfera. (Per rispetto agli astrofisici, sarebbe etimologicamente corretto definire tale spazio la “ergosfera” o “sfera del lavoro”).

In pratica però la distinzione tra noosfera ed ergosfera non riveste particolare importanza per gli obiettivi di questo saggio. È arduo verificare l'esistenza in qualche modo significativa della noosfera nel senso assoluto inteso da Fare; bisognerebbe essere un seguace di Platone per crederci davvero. E la distinzione tra noosfera ed ergosfera si rivela d'importanza *concreta* soltanto se si vuole asserire il fatto che le idee (elementi della noosfera) non possano essere posseduti, ma invece lo possono le loro manifestazioni in quanto progetti. Una questione che ci porta a tematiche riguardanti la teoria della proprietà intellettuale che vanno oltre gli scopi di questo scritto.

Per evitare confusioni è tuttavia importante notare come né la noosfera né l'ergosfera rappresentino la totalità degli spazi virtuali dei media elettronici che talvolta (tra il disgusto della gran parte degli hacker) viene definito “cyberspazio”. Qui la proprietà soggiace a regole completamente differenti, più vicine a quelle del substrato materiale – praticamente, il proprietario dei media e delle macchine sul quale è ospitato una parte del cyberspazio risulta essere proprietario anche di quest'ultimo.

La struttura proposta da Locke suggerisce chiaramente che gli hacker open source seguono quelle usanze allo scopo di difendere un qualche tipo di proventi derivanti dal loro sforzo. Tali proventi debbono dimostrarsi più significativi delle energie spese per i progetti d'insediamento, nonché dei costi per

mantenere le cronologie delle versioni in grado di documentare la “catena degli atti”, e del tempo speso per le notifiche pubbliche e per il periodo di attesa prima di re-impossessarsi di un progetto orfano.

Inoltre, la “rendita” riconosciuta dall'open source dev'essere qualcosa in più del semplice utilizzo del software, qualcos'altro che risulterebbe diluito o compromesso dai progetti di biforcazione. Se l'unica questione riguardasse il suo utilizzo, non esisterebbero tabù contro le divaricazioni progettuali, e la loro proprietà non assomiglierebbe affatto alla proprietà terriera. In realtà, è proprio questo mondo alternativo (dove l'utilizzo è l'unica rendita) l'unico elemento racchiuso nelle attuali licenze open source.

Possiamo eliminare subito alcuni candidati alla rendita. Essendo impossibile esercitare efficacemente la coercizione all'interno di una rete, va eliminata la ricerca del potere. Allo stesso modo, la cultura open source non ha niente che assomigli al denaro o a una economia interna, e quindi gli hacker non possono mirare a nulla di analogo alla ricchezza materiale.

Esiste altresì una possibilità d'arricchimento per chi opera nell'open source – un modo che potrebbe indicarne le motivazioni profonde. Occasionalmente la reputazione che si conquista nella cultura hacker può trasbordare nel mondo reale in maniera economicamente significativa. Si possono trovare lavori migliori, richieste di consulenza, contratti per scrivere libri.

Però questo tipo di effetto collaterale è raro e marginale per la maggior parte degli hacker; decisamente troppo come unica spiegazione convincente, volendo perfino ignorare le ripetute proteste degli stessi hacker secondo cui fanno quel che fanno non per denaro, ma per idealismo e dedizione.

È tuttavia il caso di prestare attenzione alla modalità secondo cui tale effetto economico collaterale viene mediato. Vedremo qui di seguito come la comprensione delle dinamiche della reputazione all'interno della *stessa* cultura open source rivesta una considerevole capacità esplicativa.

6. La cultura hacker come economia del dono

Per comprendere il ruolo della reputazione nella cultura open source, può essere d'aiuto spostarsi ulteriormente dalla storia all'antropologia e all'economia, esaminando la differenza tra le *culture dello scambio* e quelle del *dono*.

Gli esseri umani sono dotati di una predisposizione innata alla competizione per il miglioramento del proprio status sociale; qualcosa che è parte intrinseca della nostra evoluzione. Per il 90% della storia precedente all'invenzione dell'agricoltura, i nostri antenati vivevano in piccoli gruppi nomadi dediti alla caccia e alla raccolta. Agli individui con lo status sociale più alto spettavano i partner più sani e il cibo migliore. Una competizione che si esprime quindi in modi differenti a seconda del livello di scarsità delle merci necessarie alla sopravvivenza.

La gran parte delle modalità secondo cui gli esseri umani si organizzano sono adattamenti alla scarsità e alla volontà. Ogni modo porta con sé maniere differenti di ottenere il miglioramento del proprio status sociale.

Il modo più semplice è la *gerarchia di comando*. In questi casi, l'allocazione delle merci più scarse viene effettuata da un'autorità centrale sotto la minaccia della forza. Nelle gerarchie di comando c'è poco spazio per posizioni intermedie [Mal]; esse tendono a diventare brutali e inefficienti di pari passo con l'ampliamento delle dimensioni. Per questo motivo, quando superano l'ordine di grandezza della famiglia estesa, quasi sempre le gerarchie di comando si trasformano in parassiti sociali di una economia più ampia di tipo diverso. Nelle gerarchie di comando lo status sociale viene stabilito essenzialmente dall'accesso (o meno) al potere coercitivo.

La nostra società è per lo più un'*economia di scambio*. Si tratta di un sofisticato adattamento alla scarsità che, al contrario del modello del comando, funziona bene con ruoli intermedi. La distribuzione delle merci avviene in maniera decentralizzata tramite il commercio e la cooperazione volontaria (in realtà, l'effetto dominante del desiderio competitivo produce comportamenti cooperativi). In un'economia di

scambio, lo status sociale viene determinato essenzialmente dalla possibilità (o meno) di avere il controllo delle cose, non necessariamente delle cose materiali, da usare o da commerciare.

La maggior parte delle persone seguono modelli mentali impliciti relativi a entrambi i contesti, e interagiscono gli uni con gli altri basandosi su tali modelli. I governi, l'apparato militare e il crimine organizzato (per fare degli esempi) sono delle gerarchie di comando parassite della più ampia economia di scambio che chiamiamo "free market". Esiste tuttavia un terzo modello, radicalmente diverso dai primi due e generalmente non riconosciuto se non dagli antropologi: la *cultura del dono*.

Le culture del dono sono adattamenti non alla scarsità bensì all'abbondanza. Si sviluppano all'interno di popolazioni senza problemi per quanto concerne la penuria di merci necessarie alla sopravvivenza. Possiamo osservare le culture del dono in azione tra gli aborigeni che vivono in ecozone dal clima mite e cibo abbondante. Oppure in certi strati della nostra società, soprattutto nel mondo dello spettacolo e tra persone molto ricche.

L'abbondanza rende difficili le relazioni di comando e inutili le relazioni di scambio. Nelle culture del dono, lo status sociale viene determinato non da quel che si controlla ma da *quel che si regala*.

È così che avviene per la festa del potlach del capo Kwakiutl. È così che avviene per le gesta filantropiche dei multimiliardari, in genere elaborate e pubbliche. Ed è così che avviene per le lunghe ore di lavoro che occorrono all'hacker per produrre software open source di alta qualità.

Perché è ovvio che sotto quest'ottica, la società degli hacker open source non è altro che una cultura del dono. Al suo interno non esiste alcuna seria scarsità di "materiale di sopravvivenza" – spazio su disco, ampiezza di banda di rete, macchine potenti. Il software è liberamente condiviso. Un'abbondanza che crea situazioni dove l'unico ambito disponibile per la competizione sociale è la reputazione tra i colleghi.

Da sola, quest'osservazione non è però sufficiente a spiegare le caratteristiche, già illustrate, della cultura hacker. Anche i "cracker d00dz" hanno una cultura del dono che pesca nei medesimi media (elettronici), ma il loro comportamento è molto diverso. La mentalità di gruppo è molto più forte ed esclusiva che tra gli hacker. Anziché condividerli, i segreti vengono passati sotto banco; è molto comune trovare gruppi di cracker che distribuiscono degli eseguibili senza sorgenti in grado di "craccare" il software, piuttosto che consigli e dritte su come sono riusciti a farlo.

Quel che tutto ciò dimostra, nel caso non fosse ancora chiaro, è che esiste più di una maniera di far funzionare la cultura del dono. La storia e i valori perseguiti sono elementi importanti. Altrove [HH] ho già riassunto la storia della cultura hacker; le modalità seguite per dar forma alle attuali usanze non sono affatto misteriose. Gli hacker hanno definito la propria cultura tramite una serie di scelte possibili sulla forma che la competizione interna andrà prendendo. All'analisi di tale forma è dedicato il prosieguo di questo saggio.

7. La gioia dell'hacking

Nell'affrontare l'analisi del "gioco della reputazione" non intendo qui affatto svalutare o ignorare la pura soddisfazione artistica di progettare un software bellissimo e farlo funzionare. Una soddisfazione che abbiamo provato un po' tutti e che ogni volta ci fa rifiorire. Le persone per cui ciò non rappresenta una motivazione significativa, non diventeranno mai degli hacker, proprio come quanti non amano la musica non potranno mai divenire compositori.

Forse dovremmo quindi considerare un altro modello di comportamento per il quale motivazione primaria è la pura gioia dell'artigiano creativo. Secondo tale modello, le abitudini degli hacker vanno spiegate come un modo per massimizzare sia le opportunità "dell'artigianalità" sia la qualità dei risultati. È forse vero che ciò si pone in conflitto con (o suggerisce esiti diversi da) il modello del "gioco della reputazione?"

Non proprio. Analizzando il modello "dell'artigianalità", torniamo ai medesimi problemi che impongono

al regno hacker di operare come una cultura del dono. Come è possibile massimizzare la qualità se non esiste un metro per misurare quest'ultima? Se non è possibile applicare l'economia della scarsità, quali unità di misura sono disponibili oltre la valutazione dei colleghi? In definitiva sembra che ogni cultura dell'artigianalità debba auto-strutturarsi attraverso il gioco della reputazione – in realtà, possiamo osservare tali dinamiche all'interno di molti contesti storici, dalle corporazioni medievali in poi.

Sotto un aspetto importante il modello dell'artigianalità appare comunque più debole di quello della cultura del dono; da solo non basta a spiegare la contraddizione con cui abbiamo iniziato questo saggio.

Alla fin fine, la motivazione dell'artigianalità in sé può non rivelarsi così psicologicamente lontana dal gioco della reputazione come potremmo invece ritenere. Immaginiamo che quel bellissimo programma appena realizzato rimanga chiuso in un cassetto senza mai essere usato un'altra volta. E fantastichiamo invece che possa essere utilizzato efficacemente e con piacere da molta gente. Quale dei due sogni sceglieremmo?

Ciò nonostante, è il caso di tenere un occhio aperto sul modello dell'artigianalità. Intuitivamente attira molti hacker, oltre a illustrare abbastanza bene alcuni aspetti dei comportamenti individuali.

Dopo aver pubblicato la prima versione di questo saggio, ho ricevuto un commento anonimo: “Non è che lavori per farti una buona reputazione, ma se hai fatto un buon lavoro il pagamento concreto che ne deriva sono le conseguenze della buona reputazione.” Si tratta di una questione sottile e importante. È la buona reputazione a costituire un continuo incentivo a lavorare, sia che l'artigiano ne sia cosciente o meno. In definitiva, quindi, sia che un hacker comprenda o meno il proprio comportamento come parte del gioco della reputazione, sarà tale comportamento a dar forma al gioco stesso.

8. Le molte facce della reputazione

Sono diverse le ragioni generali per cui in ogni cultura del dono si tende a conquistare la buona reputazione tra i colleghi (prestigio).

Primo e più ovvio motivo, si tratta di una ricompensa essenziale. La sperimentiamo in tal modo per via delle motivazioni evolutive precedentemente descritte. (Molte persone imparano a redirigere l'interesse per il prestigio in varie sublimazioni, le quali non appaiono immediatamente inerenti al gruppo dei pari grado, quali “onore”, “integrità morale”, “pietà'”, etc.; ciò comunque non modifica il meccanismo che le sottende).

Secondo, il prestigio è una buona (e in un'economia del dono puro, l'unica) maniera per attirare attenzione e cooperazione da parte degli altri. Se qualcuno è ben noto per la sua generosità, intelligenza, saperci fare, capacità di leadership, o altre buone qualità, diventa molto più facile convincere gli altri che potranno trarre giovamento dall'associarsi con tale persona.

Terzo, se l'economia del dono è in contatto o interconnessa con un'economia di scambio o con una gerarchia di comando, la buona reputazione potrebbe propagarsi e far raggiungere uno status più elevato anche in questi contesti.

Oltre tali motivazioni d'ordine generale, sono le particolari condizioni della cultura hacker a rendere il prestigio anche più valido di quanto lo sarebbe in una cultura del dono nel “mondo reale”.

La principale tra queste condizioni particolari è che gli artefatti che si regalano risultano alquanto complessi (o, interpretato in altro modo, costituiscono i segni visibili del dono in tempo ed energia impiegate). Il loro valore non è per nulla così ovvio come quello dei doni materiali, o come il denaro alla base dell'economia di scambio. È molto più difficile fare distinzioni oggettive tra un dono sofisticato e uno grossolano. Di conseguenza, il successo sulla buona reputazione di chi dona dipende in maniera precisa dal giudizio critico dei colleghi.

Altra peculiarità è la relativa purezza della cultura open source. Gran parte delle culture del dono risultano

compromesse – per via delle relazioni in corso o con l'economia di scambio, quali il commercio in beni di lusso, oppure con l'economia del comando, tipo in famiglia o nei clan. All'interno della cultura open source non esistono analogie simili; ne consegue l'assenza pressoché totale di altre modalità per salire nella scala sociale al di fuori della buona reputazione tra colleghi.

9. Diritti di proprietà e incentivi per la reputazione

Ci troviamo ora nella posizione di portare tutte le analisi precedenti verso un contesto più generale sulle convenzioni sul tema della proprietà per come viene intesa tra gli hacker. Adesso siamo in grado di comprendere quali siano i proventi derivanti dalla colonizzazione della noosfera; si tratta della reputazione dei colleghi all'interno della cultura del dono propria degli hacker, con tutti gli effetti collaterali e le acquisizioni secondarie che ciò comporta.

Per quest'ulteriore passaggio, possiamo considerare le consuetudini relative alla proprietà nel regno degli hacker – simili alle concezioni di Locke – come uno strumento per *massimizzare gli incentivi della reputazione*; per assicurarsi cioè che i riconoscimenti collettivi vadano laddove siano dovuti e non altrove.

Alla luce di una tale analisi, i tre tabù che abbiamo evidenziato in precedenza rivestono un significato preciso. La reputazione di una persona può essere ingiustamente danneggiata se qualcun altro si appropria indebitamente del suo lavoro, o lo manomette; i seguenti tabù (e le relative convenzioni) tentano di prevenire proprio questo:.

- La divaricazione dei progetti è negativa perché espone i collaboratori precedenti al rischio di perdita della reputazione, rischio che può essere controllato soltanto qualora essi siano contemporaneamente attivi in entrambi i progetti derivanti dalla scissione. (Situazione che in genere risulta troppo confusa o difficile da gestire).
- La distribuzione di “patch” volanti (o, molto peggio, di binari volanti) espone i proprietari al rischio dell'ingiusta perdita di reputazione Anche se il codice ufficiale risulta perfetto, i proprietari subiranno rimproveri per i bug presenti nelle “patch” (vedi però [RP]).
- Cancellare surrettiziamente il nome di qualcuno da un progetto, nel contesto culturale, è uno dei crimini più gravi. Significa rubare il dono della vittima per poi offrirlo alla comunità come si trattasse di quello del ladro.

Tutti e tre questi comportamenti tabù arrecano danni globali alla comunità open source e locali alle vittime specifiche. Implicitamente danneggiano l'intera comunità facendo diminuire la percezione, da parte di ogni potenziale collaboratore, che regalando i frutti del proprio lavoro si verrà ricompensati.

È importante notare come esistano possibili spiegazioni alternative per due di questi tre tabù.

Primo, spesso gli hacker motivano l'antipatia per la scissione dei progetti lamentando l'inutile duplicazione di lavoro che ciò porterebbe, con i progetti figliati che finirebbero per seguire un'evoluzione futura più o meno parallela. Si potrebbe inoltre osservare che la biforcazione tende a dividere la comunità dei co-sviluppatori, lasciando entrambi i progetti figliati con un minor numero di cervelli da utilizzare rispetto al progetto-madre.

Qualcuno ha replicato sottolineando come sia insolito per più di un progetto derivante dalla scissione sopravvivere a lungo con una significativa “quota di mercato”. Ciò rafforza l'incentivo a partecipare ed evitare le biforcazioni per tutte le entità coinvolte, perché è troppo difficile sapere in anticipo chi si troverà dalla parte dei perdenti per vedere così gran parte del proprio lavoro scomparire o languire nell'oscurità.

Le “patch” volanti non sono gradite perché possono complicare enormemente la ricerca dei bug, aumentando così il lavoro di chi ha l'incarico della manutenzione del progetto, persone che generalmente hanno già il loro gran daffare per scovare i propri errori.

Esiste una notevole dose di verità in tali spiegazioni, e certamente servono da rinforzo per la logica della

proprietà in stile Locke. Ma pur se intellettualmente attraenti, esse non riescono a spiegare perché dovrebbe trapelare così tanta emotività e territorialità nelle rare occasioni in cui un tabù viene infranto o alterato – non soltanto dalle parti offese, bensì anche da passanti e osservatori che spesso reagiscono piuttosto duramente. Le preoccupazioni a sangue freddo sulla duplicazione del lavoro e sui problemi per la manutenzione semplicemente non possono spiegare a sufficienza tali comportamenti.

Eccoci infine al terzo tabù. È difficile immaginare cos'altro possa motivarlo se non l'analisi sul gioco della reputazione enunciata sopra. Il fatto che raramente questo tabù venga considerato appena qualcosa in più dell'affermazione “non sarebbe giusto”, è a suo modo rivelatore, come vedremo nella sezione che segue.

10. Il problema dell'ego

All'inizio di questo articolo ho scritto che la conoscenza inconscia in via di adattamento di una cultura, spesso è in contrasto con l'ideologia cosciente. Ne abbiamo già visto un esempio importante nel fatto che le consuetudini sulla proprietà di Locke siano state ampiamente imitate nonostante violassero gli intenti dichiarati nelle licenze standard.

Ho potuto osservare un altro esempio interessante di tale fenomeno discutendo con gli hacker sulla precedente analisi relativa al gioco della reputazione. Ovvero sul fatto che molti hacker si sono opposti a quell'analisi, mostrandosi assai riluttanti ad ammettere che il loro comportamento fosse motivato dal desiderio di ottenere il riconoscimento altrui, o, come l'avevo incautamente definito allora, dalla “soddisfazione egoistica”.

Questo fatto ci consente di illustrare un aspetto interessante della cultura hacker. La quale, a livello cosciente, disdegna e disprezza l'egoismo e le motivazioni egoiste. L'auto-promozione pubblicitaria viene criticata senza pietà, perfino quando la comunità potrebbe ricavarne qualcosa di positivo. Non è certo un caso che gli anziani e i “grandi” della cultura debbano sempre esprimersi sommessamente e prendersi ironicamente in giro per poter mantenere il loro status sociale. In che modo tale atteggiamento possa convivere con una struttura in apparenza fondata pressoché interamente sull'ego, è fatto che richiede energiche spiegazioni.

È certo che gran parte di tutto ciò derivi dall'atteggiamento europeo e americano verso “l'ego”, generalmente negativo. La matrice culturale della vasta maggioranza degli hacker insegna loro che il desiderio di soddisfare l'ego è qualcosa di negativo (o almeno segno d'immaturità); che al massimo l'ego è una manifestazione eccentrica tollerabile soltanto nei primi attori e spesso sintomo concreto di patologie mentali. Quel che viene generalmente accettato sono forme sublimate e camuffate, quali “reputazione tra colleghi”, “auto-stima”, “professionalità”, “vantarsi per avercela fatta”.

Potrei scrivere un intero saggio a latere sulle radici malsane di questa parte della nostra eredità culturale, e l'incredibile ammontare di danno auto-illusorio che ci infliggiamo ritenendo (contro tutte le prove psicologiche e comportamentali) che dovremmo sempre seguire motivazioni “altruiste”. E forse lo farei, se Friedrich Wilhelm Nietzsche e Ayn Rand non avessero già realizzato lavori da assoluti competenti della materia (a parte ogni ulteriore loro mancanza), procedendo alla decostruzione dell'altruismo in quanto insieme di interessi personali non ammessi.

Ma non sono qui per fare della filosofia morale o della psicologia, per cui mi limiterò a osservare un danno minore procurato dalla credenza che l'ego sia negativo, ovvero: ciò ha reso emotivamente difficile per molti hacker comprendere coscientemente le dinamiche sociali del proprio futuro!

Non abbiamo però ancora concluso con questo tipo di indagine. Il tabù della cultura circostante contro comportamenti chiaramente guidati dall'ego, risulta talmente intensificato nella (sotto)cultura hacker, che è il caso di sospettare esso possa rivestire una qualche funzione di adattamento per gli stessi hacker. È comunque evidente come questo tabù appaia più debole all'interno di molte altre culture del dono, come nel caso della gente di teatro e nei circoli dei super-ricchi.

11. Il valore dell'umiltà

Dopo aver determinato come il prestigio sia elemento centrale nei meccanismi di ricompensa della cultura hacker, abbiamo ora bisogno di comprendere perché far rimanere ciò qualcosa di semi-nascosto e ampiamente non dichiarato, è stata sempre considerata questione così centrale.

Il contrasto con la cultura "pirata" è istruttivo. Qui i modi di fare finalizzati al riconoscimento dello status sono palesi e perfino sfacciati. Questi cracker si vantano di aver distribuito "zero-day warez" (software "craccato" ridistribuito nello stesso giorno in cui si diffonde la release della versione originale), ma tengono la bocca cucita su come hanno fatto. Maghi a cui non piace far conoscere in giro i propri trucchi. Ne consegue che la conoscenza di base della cultura cracker come insieme progredisce assai lentamente.

Al contrario, nella comunità hacker, il lavoro che si fa è il proprio biglietto da visita. Esiste una meritocrazia piuttosto severa (vince la migliore opera d'artigianato), e si respira un forte ethos sul fatto che la qualità dovrebbe (meglio, deve) parlare di per sé. Il miglior applauso è quando si vede del codice che "semplicemente funziona" e facilmente riconoscibile come qualcosa di ben fatto da ogni programmatore competente. Come conseguenza, la conoscenza di base della cultura hacker progredisce molto rapidamente.

Ecco quindi che il tabù contro gli atteggiamenti egoistici fa aumentare la produttività. Ma questo è un effetto secondario; quel che qui viene direttamente protetta è la qualità dell'informazione nel sistema comunitario basato sulla valutazione dei pari grado. Ovvero, vantarsi o darsi delle arie sono atteggiamenti soppressi perché intesi come rumore che tende a coprire i segnali vitali provenienti dalle sperimentazioni in attività cooperative e creative.

All'interno della cultura hacker, il medium del dono è qualcosa di intangibile, i suoi canali di comunicazione raramente esprimono le sfumature emotive, e i contatti faccia a faccia tra i vari membri costituiscono l'eccezione anziché la regola. Ciò porta a un minor livello di tolleranza per il rumore rispetto ad altre culture del dono, e illustra molto chiaramente la pratica di pubblica umiltà richiesta agli anziani della tribù.

Il fatto di parlar piano risulta altresì funzionale quando si aspira al mantenimento di un progetto di successo; occorre convincere la comunità di possedere buone capacità di giudizio, perché gran parte dei compiti di chi si occupa del mantenimento consistono nel giudicare il codice che le persone propongono. Chi sarebbe disposto a sottomettere il proprio lavoro a qualcuno che non è chiaramente in grado di giudicare quel codice, o i cui comportamenti suggeriscono che dovranno cercare di trangugiare ingiustamente la reputazione derivante da quel progetto? I potenziali collaboratori vogliono leader dotati di sufficiente umiltà e classe da poter dire, quando oggettivamente appropriato, "Sì, questo codice funziona meglio della mia versione, lo userò" – e dare i riconoscimenti quando sono dovuti.

Un motivo ulteriore per i comportamenti umili è che raramente nel mondo open source si dà l'impressione che un progetto possa dirsi "concluso." Ciò porterebbe a una sensazione d'inutilità da parte dei possibili collaboratori. La maniera per amplificare la propria influenza è essere umili sullo stato di salute del programma. Se il fatto di vantarsi avviene tramite il codice, e poi qualcuno dice "Che diamine, non fa né x né y, perciò non è mica tanto ben fatto," spesso seguono rapide le necessarie "patch".

Infine, le mie personali osservazioni hanno rivelato che gli atteggiamenti di autocritica di alcuni hacker leader nascondono la concreta (e per nulla ingiustificata) paura di divenire oggetto di un culto della personalità. Linus Torvalds e Larry Wall hanno entrambi fornito esempi chiari e sostanziosi di tali comportamenti sfuggenti. Una sera, andando a cena con Larry Wall, scherzando gli ho detto: "Sei tu l'hacker numero uno qui – tocca a te scegliere il ristorante." Si è tirato decisamente indietro. E con piena ragione; l'incapacità di saper distinguere tra i valori condivisi e i propri leader ha portato alla rovina molte buone comunità, un percorso di cui lui e Linus non possono non essere pienamente coscienti. D'altra parte è vero che la maggior parte degli hacker vorrebbe davvero avere lo stesso tipo di problemi che ha Larry, sempre che possano essere costretti ad ammetterlo.

12. Implicazioni globali del modello del gioco della reputazione

L'analisi del gioco della reputazione presenta alcune implicazioni aggiuntive forse non del tutto ovvie. Molte di queste hanno a che fare con il fatto che si guadagna maggior prestigio dal lancio di un nuovo progetto piuttosto che dalla cooperazione in uno già esistente. Inoltre, si ottiene di più da progetti decisamente innovativi anziché da incrementi proporzionali su software già in circolazione. D'altra parte, quei programmi comprensibili o utili solo all'autore sono del tutto inutili per il gioco della reputazione, e spesso risulta più semplice attirare l'attenzione contribuendo a un progetto esistente piuttosto che far notare alla gente il lancio di uno nuovo. Infine, è molto più difficile competere con un progetto che ha già ha avuto successo che non riempire una nicchia vuota.

Esiste insomma una distanza ottimale dai propri vicini (i progetti rivali più simili). Avvicinandosi troppo, il prodotto dimostra poco valore, si manifesta come un dono scadente (meglio allora collaborare a un progetto in corso). Allontanandosi troppo, nessuno potrà usarlo, né comprendere o percepire la rilevanza degli sforzi personali (di nuovo, un dono scadente). Tutto ciò crea un percorso d'insediamento nella noosfera che assomiglia parecchio a quello dei pionieri sulla frontiera geografica – non esattamente casuale, piuttosto una sorta di un'ondata frattale a diffusione limitata. I progetti tendono ad attivarsi per riempire vuoti funzionali vicino alla linea di frontiera.

Alcuni progetti molto ben riusciti sono partiti come “killer di categoria”; nessuno vuole insediarsi nei loro pressi poiché risulterebbe troppo difficile competervi onde ottenere l'attenzione degli hacker. Anziché vedere naufragare inesorabilmente i propri sforzi, sarebbe preferibile invece aggiungere delle estensioni a questi progetti, importanti e riusciti. Il classico esempio di “killer di categoria” è Emacs GNU; le sue varianti riempiono così perfettamente la nicchia ecologica riservata a un editor completamente programmabile, che nessuno ha neppure abbozzato un design veramente diverso fin dai primi anni '80. Al contrario, la gente si mette a scrivere altre modalità Emacs.

Globalmente, nel corso del tempo le due tendenze (riempimento di vuoti e killer di categoria) hanno portato a prevedibili filoni nell'avvio dei progetti. Negli anni '70 gran parte dell'open source esistente non rappresentava altro che giochetti e demo. Il decennio successivo vide la spinta verso lo sviluppo e gli strumenti per Internet. Negli anni '90 l'attenzione si è rivolta ai sistemi operativi. In ognuno di questi casi, una volta quasi esaurite le possibilità del filone precedente, venne attaccato un nuovo e più difficile livello di problemi.

Una tendenza che rivela interessanti implicazioni per il futuro prossimo venturo. All'inizio del 1998, Linux sembrava proprio il killer di categoria per la nicchia “sistemi operativi open source” – coloro che altrimenti avrebbero realizzato possibili sistemi rivali, ora scrivono invece driver ed estensioni per Linux. Senza contare come già esistano innumerevoli strumenti di basso livello, in quantità anche superiore a quel che la cultura avesse mai immaginato di possedere in quanto open source. Cos'è che manca, dunque?

Le applicazioni. Con l'avvicinarsi dell'anno 2000 sembra certo prevedere che le energie di sviluppo open source si rivolgeranno sempre più verso l'ultimo territorio vergine – programmi per non-informatici. Un chiaro indicatore iniziale arriva dallo sviluppo di GIMP, il laboratorio d'immagini tipo Photoshop, la quale rappresenta la prima maggiore applicazione open source dotata del tipo di interfaccia GNU user-friendly, considerata de rigeur nelle applicazioni commerciali dell'ultima decade. Un'altra indicazione viene dal gran parlare che si fa intorno a progetti di pacchetti applicativi quali KDE e GNOME.

Infine, l'analisi del gioco della reputazione bene illustra la frequente citazione secondo cui non si diventa hacker semplicemente autodefinendosi tali – bensì quando si viene così chiamati dagli *altri hacker*. Alla luce di questa considerazione, un “hacker” è qualcuno che ha dimostrato (contribuendo con dei doni) di possedere sia abilità tecnica sia comprensione del gioco della reputazione. Una posizione che deriva soprattutto da coscienza e crescita culturale, e che può essere riconosciuta soltanto da coloro che fanno già parte integrante di quella cultura.

13. Proprietà noosferica e etologia del territorio

Per meglio comprendere le conseguenze relative alle convenzioni sulla proprietà, ci sarà d'aiuto considerarle ancora da un'ulteriore angolo visuale: quello dell'etologia animale, in particolare l'etologia

del territorio.

La proprietà è un'astrazione delle territorialità del regno animale, evolutasi a riduzione della violenza intra-specie. Contrassegnando i propri confini, e rispettando quelli degli altri, un lupo diminuisce le probabilità di essere coinvolto in una zuffa da cui potrebbe uscire indebolito (o finanche ucciso), sminuendo così le proprie capacità riproduttive.

Allo stesso modo, la funzione della proprietà nelle società umane è quella di prevenire i conflitti inter-umani stabilendo quei confini che separano con chiarezza i comportamenti pacifici dalle aggressioni. Talvolta appare corretto descrivere la proprietà umana come una convenzione sociale arbitraria, ma avremmo imboccato una strada senz'uscita. Chiunque abbia mai avuto un cane che abbaiava agli estranei che si avvicinavano alla proprietà del padrone, ha sperimentato la continuità tra la territorialità animale e la proprietà umana. Su questo i cugini addomesticati del lupo sono istintivamente più intelligenti di molti buoni teorici politici.

L'affermazione della proprietà (come la definizione del territorio) è un atto che viene reso manifesto, un modo di dichiarare i confini che saranno difesi. Il sostegno della comunità all'affermazione della proprietà rappresenta una maniera di minimizzare la frizione e massimizzare i comportamenti cooperativi. Tutto ciò rimane vero pur quando "l'affermazione della proprietà" appare molto più astratta di una rete di recinzione o dell'abbaiare di un cane, perfino quando si tratta soltanto della presa di posizione di chi si occupa del mantenimento di un progetto nel file README. Si tratta pur sempre di un'astrazione della territorialità, e (al pari di altre forme di proprietà) i nostri modelli di proprietà basati sull'istinto non sono altro che l'evoluzione di quelli territoriali, il cui scopo rimane quello di favorire la risoluzione dei conflitti.

Ad una prima occhiata, quest'analisi etologica sembra molto astratta e difficile da collegare al comportamento concreto degli hacker. Ma presenta alcuni risvolti importanti. Uno sta nella spiegazione della popolarità acquisita dai siti sul World Wide Web, e soprattutto nel perché i progetti open source dotati di siti Web appaiono molto più "reali" e sostanziali di quelli che ne sono sprovvisti.

Considerato in maniera oggettiva, ciò sembra difficile da spiegare. Paragonata alle energie necessarie per lanciare e mantenere un programma anche piccolo, una pagina Web è qualcosa di semplice; risulta pertanto fuori luogo considerarla prova evidente di sforzi insoliti o sostanziali.

Neppure le caratteristiche funzionali del Web costituiscono una spiegazione sufficiente. Le funzionalità comunicative di una pagina Web possono essere ugualmente, o anche in maniera più efficace, attivate tramite la combinazione di un sito FTP, una mailing list, e l'inserimento di testi su Usenet. In realtà è piuttosto insolito che le comunicazioni di routine di un progetto avvengano tramite il Web anziché via mailing list o newsgroup. Perché, allora, la popolarità dei siti Web come dimora, "home", dei progetti?

È l'implicita metafora del termine "home page" a fornire un primo importante suggerimento. Mentre l'avvio di un progetto open source rappresenta un'affermazione territoriale nella noosfera (e tale riconosciuta dalle convenzioni), sul piano psicologico non risulta terribilmente avvincente. Dopo tutto, il software non dimora in luoghi naturali ed è riproducibile all'istante. E se è assimilabile alle nozioni istintive di "territorio" e "proprietà", ciò avviene soltanto dopo non pochi sforzi.

Un progetto su una home page dà concretezza all'insediamento astratto nello spazio dei possibili programmi, esprimendo come "home" quel territorio all'interno del regno suddiviso in spazi rappresentato dal World Wide Web. Il fatto di scendere dalla noosfera al "cyberspazio", non ci fa ancora raggiungere il mondo reale delle reti di recinzioni e dei cani cha abbaiano, ma aggancia in maniera più sicura l'astratta affermazione di proprietà al nostro atto istintivo di recintare il territorio. Ecco perché i progetti dotati di pagine Web appaiono più "reali."

Inoltre, una tale analisi etologica ci incoraggia a guardare più da vicino quei meccanismi atti alla gestione dei conflitti all'interno della cultura open source. Da quanto esposto, potremmo attenderci che, insieme alla massimizzazione degli incentivi per la reputazione, le abitudini relative alla proprietà debbano

rivestire un ruolo importante anche nella prevenzione e nella risoluzione dei conflitti.

14. Cause di conflitto

Sono riconducibili a quattro le maggiori aree di conflitto all'interno del software open source:

- Chi prende le decisioni vincolanti su un progetto?
- A chi addossare meriti e demeriti per cosa?
- Come ridurre la duplicazione degli sforzi ed evitare che versioni volanti complichino la ricerca dei bug?
- Tecnicamente parlando, qual è la giusta cosa da fare?

Se diamo per un momento un'occhiata all'ultima questione, tuttavia, essa tende a sparire. Come premessa, occorre stabilire l'esistenza, o meno, di un modo oggettivo di decidere se questa giusta cosa venga accettata da tutte le entità coinvolte. Qualora ciò non sia possibile, la domanda si riduce a "chi decide?"

Di conseguenza, le tre possibili cause di conflitti da risolvere riguardo un progetto sono: (A) dove ci si ferma per le decisioni relative al design, (B) in che modo decidere quali collaboratori prenderanno il merito e secondo quali modalità, (C) come far sì che il gruppo e il prodotto non si scompongano in più diramazioni.

Nella risoluzione dei punti (A) e (C) appare evidente il ruolo delle consuetudini in tema di proprietà. Le usanze sostengono che tocca ai proprietari del progetto prendere le decisioni vincolanti. Abbiamo già osservato come tali usanze esercitino forti pressioni contro lo stemperamento della proprietà tramite la scissione dei progetti.

È interessante notare l'appropriatezza di tali convenzioni anche se qualcuno dovesse dimenticare il gioco della reputazione, riesaminandole all'interno del modello puramente "artigianale" della cultura hacker. In questo caso, le consuetudini hanno meno a che fare con la diluizione degli incentivi per la reputazione e sono invece più tese a proteggere il diritto dell'artigiano di portare a termine la sua visione secondo le proprie modalità operative.

Tuttavia, il modello artigianale non è di per sé sufficiente a spiegare le convenzioni hacker riguardo il punto (B), chi prende il merito e per cosa (perché un artigiano puro, disinteressato al gioco della reputazione, non avrebbe motivo di preoccuparsene). Per analizzare la questione, dobbiamo portare un passo più avanti la teoria di Locke, passando in esame i conflitti e le operazioni relative ai diritti di proprietà sia all'interno dei progetti stessi sia tra di loro.

15. Struttura e proprietà di un progetto

Il caso più ovvio è quando esiste un'unica persona che ha la proprietà o il mantenimento del progetto. È costui che prende ogni decisione e gestisce meriti e demeriti. Gli unici conflitti possibili nascono dalle questioni sulla successione – chi sarà il nuovo proprietario se il vecchio sparisce o non ha più interesse. Occorre tener conto anche della necessità della comunità, secondo il punto (C), nel prevenire potenziali biforcazioni del progetto. Interessi espressi nella norma culturale per cui chi ne è proprietario/mantenitore, nel caso non possa più seguire il progetto, dovrebbe passarne pubblicamente la qualifica a qualcuno che ritenga adatto.

Il caso più semplice delle situazioni meno ovvie è quando un progetto ha diverse persone che si occupano del mantenimento, i quali lavorano alle direttive di un "dittatore amichevole" che è proprietario del progetto. Le consuetudini favoriscono tali circostanze per progetti ampi come il kernel di Linux o Emacs, e risolvono il problema del "chi decide" in maniera nient'affatto peggiore di ogni altra possibile alternativa.

Tipicamente è un'organizzazione a dittatura amichevole quella che emerge quando il fondatore attira dei collaboratori. Pur restando una sorta di dittatore, il proprietario introduce un nuovo livello di possibili

dispute sulla scelta di chi deve avere il merito e per quale parte del progetto.

In questa situazione, le usanze impongono al proprietario/dittatore l'obbligo di assegnare correttamente i meriti (per esempio, tramite le appropriate menzioni nel file contenente la cronologia delle versioni o nel README). Secondo il modello di proprietà di Locke, ciò significa che contribuendo a un progetto, in cambio ci si merita parte della reputazione complessiva (positiva o negativa).

Seguendo questa logica, vediamo che in realtà il dittatore amichevole non detiene l'intero progetto senza essere in possesso delle necessarie qualifiche. Pur avendo il diritto di prendere le decisioni vincolanti, in effetti egli commercia varie parti della reputazione complessiva in cambio del lavoro altrui. L'analogia con la condivisione del raccolto in un podere è quasi irresistibile, ad eccezione del fatto che il nome del collaboratore rimane nei “credit” e continua a “guadagnare” qualcosa anche quando non svolge più alcuna parte attiva nel progetto.

Quando nuovi partecipanti si aggiungono ai progetti gestiti da dittatori amichevoli, questi ultimi tendono a dividere in due filoni i collaboratori: quelli ordinari e i co-sviluppatori. Il percorso tipico per divenire co-sviluppatore è assumere la responsabilità di un sottosistema importante del progetto. Un altro è assumere il ruolo del “grande sistematore”, ovvero di chi mette a fuoco e sistema i vari bug. In un modo o nell'altro, i co-sviluppatori sono coloro che investono del tempo nel progetto in maniera sostanziale e continuata.

Il ruolo del proprietario di sottosistema è particolarmente importante per la nostra analisi e merita ulteriori approfondimenti. Agli hacker piace dire che “l'autorità fa seguito alla responsabilità”. Un co-sviluppatore che accetta la responsabilità per la gestione di un certo sottosistema generalmente arriva a controllare sia l'implementazione di quest'ultimo sia l'interfaccia con il resto del progetto, soggetto soltanto a correzioni da parte del leader (che agisce a mo' di architetto). È il caso di notare come questa regola in pratica produca una serie di proprietà chiuse all'interno di un progetto, secondo il modello di Locke, e svolga esattamente la medesima funzione di prevenzione dei conflitti ricoperta da altri tipi di confini a difesa della proprietà.

Secondo le convenzioni, ci si aspetterebbe che il “dittatore” o il leader in un progetto cui collaborano altre persone, le consulti in occasione di decisioni chiave. Ciò soprattutto quando la decisione riguarda un sottosistema di “proprietà” di un co-sviluppatore (il quale, per meglio dire, vi ha investito tempo e ne ha assunto la responsabilità). Un leader sapiente, riconoscendo la funzione dei confini per le proprietà interne del progetto, non interferirà minimamente né modificherà le decisioni prese dai proprietari di sottosistemi.

Alcuni progetti molto ampi non consentono piena aderenza al modello del “dittatore amichevole”. Un modo di superarlo è trasformare i co-sviluppatori in un comitato votante (è il caso di Apache). Un altro è la rotazione della funzione di leader, dove il controllo viene occasionalmente trasferito da un membro all'altro all'interno della cerchia dei co-sviluppatori anziani (gli sviluppatori Perl si organizzano così).

Queste complicate aggregazioni vengono considerate del tutto instabili e difficili. Chiaramente si tratta di una difficoltà che per lo più fa parte dei noti pericoli insiti nell'organizzazione di un “comitato di design”, oltre che nei comitati in quanto tali; si tratta di problemi che la cultura hacker comprende in piena coscienza. Credo tuttavia che parte del viscerale sconforto che gli hacker provano rispetto ai comitati o alla rotazione dei leader, derivi dal fatto che questi casi entrino con difficoltà in quel modello di Locke utilizzato inconsciamente dagli hacker per risolvere le questioni più semplici. Quando le organizzazioni si fanno così complesse, risulta problematico verificare le funzionalità della proprietà sia nel senso del controllo sia del ritorno di reputazione. È difficile vedere con esattezza la posizione dei confini interni, e quindi altrettanto difficile evitare i conflitti, a meno che il gruppo non goda di un livello eccezionalmente alto di armonia e fiducia reciproca.

16. Conflitti e risoluzioni

Abbiamo visto come all'interno dei progetti, la crescente complessità dei ruoli venga espressa tramite la

distribuzione dell'autorità per il design e dei parziali diritti di proprietà. Mentre ciò appare un modo efficace per la circolazione degli incentivi, al contempo provoca la diluizione dell'autorità del leader – fatto ancor più importante, tende a stemperare la sua autorità nei frangenti relativi al blocco di potenziali conflitti.

Pur se le questioni tecniche relative al design sembrano costituire i maggiori rischi per conflittualità interne, raramente questi casi portano a seri litigi. In genere vengono facilmente risolti seguendo la regola territoriale per cui l'autorità fa seguito alla responsabilità.

Un altro modo di risoluzione dei conflitti è per anzianità – nel caso di disputa obiettivamente irrisolvibile tra due collaboratori o gruppi di collaboratori, dove nessuno dei due sia proprietario del territorio in cui essa avviene, vince la parte che ha dedicato maggior tempo e lavoro al progetto nel suo insieme (ovvero, la parte che detiene più diritti di proprietà nell'intero progetto).

In genere queste regole risultano sufficienti per risolvere la maggior parte delle dispute relative a un progetto. Quando ciò non sia possibile, normalmente è l'autorità del leader a risolvere la questione. Sono assai rari i conflitti che sopravvivono a entrambi questi filtri.

Normalmente i conflitti non divengono faccenda seria a meno che questi due criteri ("l'autorità fa seguito alla responsabilità" e "vince il più anziano") tendano verso direzioni opposte, mentre l'autorità del leader risulta debole o assente. Il caso più comune in cui ciò può verificarsi è una disputa per la successione che insorge a seguito della scomparsa del leader del progetto. Mi è capitato di trovarmi coinvolto in un litigio di questo tipo. È stato qualcosa di amaro, penoso, interminabile, risoltosi soltanto quando tutte le parti coinvolte furono talmente esauste da decidere di passare volentieri il controllo a una persona esterna. Spero devotamente di non dovermi mai più trovare in qualcosa di simile.

In conclusione, tutti questi meccanismi per la risoluzione dei conflitti poggiano sulla volontà di applicarli da parte della comunità hacker nel suo insieme. Gli unici meccanismi a disposizione per imporli sono le "flame" e l'allontanamento – condanne pubbliche di quanti hanno infranto le convenzioni, e successivo rifiuto a cooperare ulteriormente con loro.

17. Meccanismi d'acculturazione e connessioni con il mondo accademico

Una prima versione di questo saggio si poneva la domanda: Come fa la comunità a informare e istruire i propri membri sulle consuetudini in vigore? Sono forse queste auto-evidenti o auto-organizzate a livello semi-inconscio, vengono insegnate tramite l'esempio oppure seguendo specifiche istruzioni?

Va detto che raramente si ricorre a queste ultime, non foss'altro perché da tempo esistono alcune descrizioni esplicite delle norme in vigore nella cultura ancor'oggi attive.

Molte norme vengono insegnate tramite l'esempio. Per citare un caso assai semplice, esiste una regola per cui ogni distribuzione di software dovrebbe contenere il file README o READ.ME, che a sua volta riporta le prime istruzioni relativa alla distribuzione stessa. Una convenzione stabilita e applicata almeno dai primi anni '80, pur non essendo mai stata scritta esplicitamente. La si deduce dando un'occhiata alle molte distribuzioni avvenute.

D'altra parte alcune consuetudini hacker si auto-organizzano da sole una volta acquisita la comprensione di base (forse inconscia) del gioco della reputazione. Alla maggior parte degli hacker non è mai stato insegnato nulla riguardo i tre tabù elencati nel terzo capitolo, o comunque li riterrebbero auto-evidenti piuttosto che trasmessi. Questo fenomeno richiede un'analisi più ravvicinata – e forse possiamo trovarne spiegazione seguendo il processo secondo il quale gli hacker acquisiscono le conoscenze intrinseche della cultura.

Molte culture fanno uso di indizi nascosti (più precisamente, "misteri", in senso mistico-religioso) come meccanismi di acculturazione. Si tratta di segreti non rivelati agli estranei, ma che ci si aspetta vengano scoperti o dedotti dagli aspiranti newbie. Per essere accettati, occorre dimostrare sia di comprendere tali

misteri sia di averli assimilati in modo culturalmente corretto.

La cultura hacker fa un uso insolitamente cosciente ed esteso di tali indizi o test. Un processo che è possibile osservare almeno a tre diversi livelli:

- Misteri espliciti simili alle password. Come esempio, c'è un newsgroup di USENET chiamato alt.sysadmin.recovery che contiene uno di tali segreti molto espliciti; non è possibile inserirvi dei testi senza conoscerlo, e il fatto di esserne a conoscenza è la prova che si è in grado di farlo. Un segreto che gli habitué hanno un forte tabù a rivelare.
- Requisiti necessari per l'iniziazione in certi misteri tecnici. Occorre assorbire una buona quantità di conoscenze tecniche prima di poter offrire doni di valore (occorre, ad esempio, conoscere almeno uno dei più importanti linguaggi informatici). Questo requisito funziona come per gli indizi nascosti ma su scala più grande, ovvero come filtro per le qualità necessarie a muoversi correttamente all'interno della cultura (tra cui: capacità di pensiero astratto, persistenza, flessibilità mentale).
- Misteri d'ambito sociale. Per essere coinvolti nella cultura occorre legarsi a progetti specifici. Ognuno di questi rappresenta un contesto sociale degli hacker che il futuro collaboratore deve investigare e comprendere, a livello sia sociale sia tecnico, per poter funzionare. (Concretamente, il modo comune di farlo è leggendo le pagine Web del progetto e/o gli archivi email). È attraverso questi progetti di gruppo che i newbie sperimentano gli esempi comportamentali degli hacker più esperti.

Nel processo di acquisizione di tali misteri, l'hacker potenziale apprende la conoscenza contestuale che (dopo qualche tempo) rende “auto-evidenti” i tre tabù e le altre consuetudini.

Incidentalmente, è possibile sostenere che la struttura della stessa cultura del dono degli hacker sia dotata di un proprio mistero centrale. Non si è considerati acculturati (in concreto: nessuno ti chiamerà hacker) fino a quando non si dimostri comprensione a livello profondo del gioco della reputazione e dei suoi impliciti usi, costumi e tabù. Ma si tratta di elementi ovvi; ogni cultura richiede tali comprensioni dai futuri membri. Inoltre, la cultura hacker non dimostra alcun desiderio di mantenere segrete le proprie logiche interne e le usanze popolari – o, almeno, nessuno, mi ha insultato mai per averle divulgate!

Parecchi commenti, troppo numerosi da elencare, a seguito di questo saggio hanno rilevato il fatto che le usanze degli hacker sulla proprietà appaiono intimamente connesse a (e possono derivare direttamente da) certe pratiche tipiche del mondo accademico, in particolare nella comunità dei ricercatori scientifici. Quest'ultima presenta problemi analoghi per quanto concerne l'insediamento sul territorio di idee potenzialmente produttive, e offre soluzioni molto simili nel senso della reputazione e della revisione da parte dei pari grado.

Poiché molti hacker hanno avuto contatti e formazione con il mondo accademico (è normale imparare a effettuare l'hacking dei programmi negli anni del college), i comuni percorsi d'adattamento tra accademia e cultura hacker rappresentano un contesto ben più che casuale per comprendere le modalità applicative di tali convenzioni.

Il mondo accademico è ricco di ovvi paralleli con la “cultura del dono” degli hacker per come l'ho finora caratterizzata. Una volta ottenuta la cattedra di ruolo, il ricercatore non ha più alcun bisogno di preoccuparsi per la propria sopravvivenza (infatti, il concetto stesso di posto fisso può essere probabilmente ricondotto a una precedente cultura del dono nella quale i “filosofi naturali” erano innanzitutto signori benestanti con parecchio tempo a disposizione da devolvere alla ricerca). In mancanza di preoccupazioni per il pane quotidiano, la meta trainante diviene il miglioramento della reputazione personale, che incoraggia la condivisione di idee e ricerche tramite le pubblicazioni e altri media. Ciò è perfettamente funzionale poiché la ricerca scientifica, al pari della cultura hacker, si basa ampiamente sul concetto che occorra “poggiarsi sulle spalle dei giganti”, non dovendo cioè riscoprire ogni volta i principi di base da cui muovere, ma utilizzando e incrementando quanto già fatto da altri.

Alcuni si sono spinti fino a suggerire che le convenzioni hacker non rappresentano altro che un riflesso

della usanze popolari in vigore nella comunità dei ricercatori, e che in pratica (per la gran parte) vengono colà acquisite. Con tutta probabilità si tratta di una stima in eccesso, non foss'altro perché sono proprio certe usanze hacker a essere rapidamente assorbite dai ricercatori più intelligenti!

In realtà qui ci troviamo di fronte a una possibilità più interessante. Nutro il sospetto che il mondo accademico e la cultura hacker condividano percorsi d'adattamento non perché geneticamente collegate, ma in quanto entrambe hanno dato vita a un'organizzazione sociale ottimale rispetto a quanto stavano cercando di fare, alle leggi naturali e all'istintivo relazionarsi proprio degli esseri umani. Il verdetto della storia sembra essere che il capitalismo del libero mercato rappresenti la maniera ottimale a livello globale per cooperare verso l'efficienza economica; forse, parimenti, la cultura del dono e il gioco della reputazione costituiscono il modo ottimale a livello globale per cooperare verso la produzione (e la verifica!) di lavoro creativo di alta qualità.

Se ciò dovesse risultare vero, direi che questo punto si rivelerebbe ben più che d'interesse accademico. Sembrerebbe suggerire, da un angolo di visuale leggermente diverso, una delle speculazioni già tracciate in La cattedrale e il bazaar; quella che alla fin fine la modalità industriale-capitalista della produzione del software fosse destinata a essere affossata fin da quando il capitalismo stesso iniziò a creare sufficiente surplus di ricchezza per molti programmatori, consentendo loro di vivere nella cultura del dono successiva alla scarsità.

18. Conclusione: dalla convenzione alla legge convenzionale

Abbiamo esaminato le consuetudini che regolano la proprietà e il controllo del software open source. Abbiamo visto come queste sottendono dei diritti di proprietà omologhi alla teoria di Locke sulla proprietà terriera. Abbiamo collegato ciò con l'analisi della cultura hacker in quanto “cultura del dono”, dove i partecipanti competono per il prestigio regalando tempo, energia, creatività. Abbiamo esaminato le implicazioni di tale analisi per la risoluzione dei conflitti all'interno di questa cultura.

Logicamente la prossima domanda da porsi è “Perché tutto ciò dovrebbe essere importante?” Gli hacker hanno sviluppato siffatte consuetudini senz'alcuna analisi cosciente e ugualmente senz'alcuna analisi cosciente le hanno seguite (fino ad ora). A questo punto non appare immediatamente chiaro se, e come, l'analisi cosciente possa farci guadagnare qualcosa di pratico – a meno che, forse, non ci si sposti dalla descrizione alla prescrizione, onde poter trarre modalità di miglioramento nella funzionalità di tali convenzioni.

Abbiamo trovato un'analogia logica ravvicinata per le usanze hacker nella teoria della proprietà terriera tipica della tradizione angloamericana del diritto consuetudinario. Storicamente [Miller], le culture tribali europee che diedero vita a questa tradizione hanno migliorato i propri sistemi di risoluzione dei conflitti passando da un insieme di usanze semi-coscienti, non articolate, a un corpo di leggi convenzionali memorizzate dai saggi della tribù – e infine scritte.

Forse, con l'aumento della nostra popolazione e la maggiore difficoltà di acculturazione dei nuovi membri, è giunta l'ora per la cultura hacker di apprestarsi a qualcosa di analogo – mettere a punto dei codici scritti ove indicare le pratiche necessarie alla risoluzione delle varie dispute che possano sorgere in connessione con i progetti open source, e dar corpo a una tradizione di arbitrariato dove ai membri più anziani possa esser chiesto da fungere da mediatori sulle dispute in corso.

L'analisi contenuta in questo saggio delinea le linee-guida su cui potrebbe basarsi il possibile codice, rendendo in maniera esplicita quel che precedentemente era implicito. Nulla di quanto incluso in tale codice potrebbe venire imposto dall'alto; dovrebbe invece essere volontariamente adottato dai fondatori o dai proprietari dei progetti individuali. Né potrebbe trattarsi di leggi rigidamente intese, poiché è probabile che le pressioni sulla cultura tendano a modificarsi nel corso del tempo. Infine, per consentire la corretta applicazione di un tale codice, questo dovrebbe ottenere l'ampia approvazione della tribù degli hacker.

Ho iniziato a lavorare alla stesura di queste leggi convenzionali, sotto il titolo provvisorio di “Malvern

Protocol", dal nome della cittadina in cui vivo. Se le analisi generali presentate in questo saggio dovessero ricevere ampia accettazione, è mia intenzione mettere a disposizione di tutti il Malvern Protocol come codice-modello per la risoluzione delle dispute. Chiunque fosse interessato a criticare e a sviluppare questo documento, è invitato a contattarmi via email, come pure per quanti vogliano inviare feedback o soltanto farmi sapere se si tratta di una buona idea o meno.

19. Domande per ulteriori approfondimenti

La comprensione da parte della cultura (oltre che del sottoscritto) dei grandi progetti che non seguono il modello del dittatore amichevole è ancora scarsa. La maggior parte di questi progetti restano bloccati. Alcuni hanno ottenuto un successo spettacolare e importante (Perl, Apache, KDE). Nessuno riesce davvero a capire dove sia la differenza.

(Esiste la vaga sensazione che si tratti di progetti sui generis che prosperano o decadono soltanto in virtù della dinamica di gruppo attivata da quei collaboratori specifici: ma è vero oppure esistono strategie replicabili che un gruppo possa efficacemente seguire?)

Come questione concreta osservabile, le persone che trovano e lanciano progetti di successo raccolgono maggior prestigio di quelli che svolgono la stessa quantità di lavoro nel debugging e nell'assistenza a progetti di successo. Si tratta di una valutazione razionale delle energie comparative, oppure di un effetto secondario del modello territoriale inconscio sopra citato?

20. Bibliografia, note e ringraziamenti

[*Miller*] Miller, William Ian; *Bloodtaking and Peacemaking: Feud, Law, and Society in Saga Iceland*; University of Chicago Press 1990, ISBN 0-226-52680-1.

Un affascinate studio sulle leggi popolari dell'Islanda, che getta luce sui primordi della teoria di Locke e descrive i passaggi successivi del processo storico tramite il quale le convenzioni si sono trasformate in leggi convenzionali e da qui in leggi scritte.

[*Mal*] Malaclypse the Younger; *Principia Discordia, or How I Found Goddess and What I Did To Her When I Found Her*; Loompanics, ISBN 1-55950-040-9.

In mezzo a molte illuminanti sciocchezze, il cosiddetto "principio dello SNAFU" fornisce un'analisi piuttosto acuta sul perché nelle gerarchie di comando ci sia poco spazio per posizioni intermedie. Ne esiste anche una versione HTML.

[*BCT*] J. Barkow, L. Cosmides, and J. Tooby (Eds.); *The adapted mind: Evolutionary psychology and the generation of culture*. New York: Oxford University Press 1992. Eccellente introduzione alla psicologia evolutiva. Alcuni dei saggi presenti rimandano direttamente alle tre tipologie culturali da me discusse (comando/scambio/dono), suggerendo come questi percorsi siano profondamente radicati nella psiche umana.

[*MHG*] Goldhaber, Michael K.; *The Attention Economy and the Net*.

Ho scoperto questo testo online dopo la mia versione 1.7. Pur contenendo evidenti falle (la tesi di Goldhaber sull'inapplicabilità delle motivazioni economiche nei confronti dell'attenzione non viene esaminata con la dovuta attenzione), l'autore dice cose divertenti e interessanti sul ruolo della ricerca dell'attenzione nei contesti organizzativi. In tal senso, il prestigio o la reputazione tra colleghi di cui sopra può essere considerato un caso particolare di ricerca dell'attenzione.

[*HH*] Il mio riassunto sul regno degli hacker è qui reperibile. Il libro che lo illustra davvero bene deve ancora essere scritto, probabilmente non dal sottoscritto.

[*N*] "Noosfera" è un termine oscuro in uso nella filosofia dell'arte, derivato dal greco "nous", ovvero

mente, spirito o anche respiro. La prima parte ("noo") suona praticamente uguale all'inglese "know" (radice di "conoscenza, sapere"). Volendo essere pignoli sull'ortografia, ci vorrebbe una diaresi sopra una delle due "o" – non chiedetemi però quale.

[RP] Vanno segnalate alcune sottigliezze sulle "patch" volanti. Queste possono essere suddivise in "amichevoli" e "non amichevoli". Quelle del primo tipo sono progettate per fondersi nei sorgenti principali del progetto, sotto il controllo di chi si occupa della manutenzione (sia che tale fusione avvenga poi effettivamente o meno); quelle "non amichevoli" sono pensate per far andare il progetto in una direzione disapprovata da chi si occupa della manutenzione. Alcuni progetti (incluso lo stesso kernel di Linux) sono piuttosto aperti ai "patch amichevoli", incoraggiandone perfino la distribuzione indipendente in quanto parte della fase di beta-test. Una "patch non amichevole", per contro, rappresenta la decisione di voler entrare in competizione con l'originale ed è una questione seria. Mantenere un'intera sequenza di queste ultime, significa tendere alla biforcazione del progetto.

Sono in debito con Michael Funk per aver sottolineato la positività del contrasto tra la cultura hacker e quella "pirata". Robert Lanphier ha contribuito molto alla discussione sui comportamenti privi di ego. Eric Kidd ha messo in luce il ruolo della valutazione dell'umiltà nella prevenzione del culto della personalità. Il capitolo sugli effetti globali ha tratto ispirazione dai commenti di Daniel Burn. Mike Whitaker ha invece ispirato la tesi portante della sezione sull'acculturazione.

21. Storia delle varie versioni

Unico responsabile per quanto appare in questo saggio è il sottoscritto, inclusi errori ed equivoci. Ho comunque integrato critiche e commenti ricevuti, sempre benvenuti, in un processo continuo che non credo possa concludersi entro uno predeterminato periodo di tempo.

10 aprile 1998: pubblicata sul Web la versione 1.2.

12 aprile 1998: versione 1.3. Correzione dei refusi e inserimento delle repliche al primo giro di commenti pubblici. Prime quattro voci della bibliografia. Aggiunta l'osservazione anonima sugli incentivi per la reputazione che funzionano anche quando l'artigiano non ne sia cosciente. Aggiunto l'istruttivo contrasto con "warez d00dz", il materiale sulla premessa "il software dovrebbe parlare di per sé", e le osservazioni su come evitare il culto della personalità. In conseguenza di queste modifiche, il capitolo sui "problemi dell'ego" risulta ampliato e diviso in due parti.

16 aprile 1998: versione 1.7. La nuova sezione sulle "implicazioni globali" discute le tendenze storiche nella colonizzazione della noosfera, e prende in esame il fenomeno dei "killer di categoria". Aggiunta un'altra domanda per ulteriori approfondimenti.

27 aprile 1998: versione 1.8. Aggiunto alla bibliografia il testo di Goldhaber. Questa la versione che apparirà negli atti pubblicati alla Linux Expo.

26 maggio 1998: versione 1.9. Incorporata la distinzione noosfera/ergosfera di Fare Rideau. Incorporata la posizione non anticommerciale di Richard Stallman. Nuova sezione su acculturazione e accademia (grazie a Ross J. Reedstrom, Eran Tromer, Allan McInnes, e altri). Aggiunte sul tema dell'umiltà ("comportamenti privi di ego") proposte da Jerry Fass e Marsh Ray.

11 luglio 1998: versione 1.10. Dietro suo suggerimento, rimosso il riferimento di Fare Rideau alla "fama".

21 novembre 1998: versione 1.14. Minori aggiustamenti editoriali e aggiornamenti dei link.

Traduzione italiana a cura di Bernardo Parrella, Giugno 1999

NdT: per snellezza del testo, si è scelto di tradurre i pronomi e i riferimenti neutri in inglese con il solo maschile italiano (autore, proprietario, etc.); tutti questi casi, vanno ovviamente intesi anche al femminile;

Testo originale inglese: http://www.tuxedo.org/~esr/writings/homesteading/homesteading.html